



Prof. Philip Koopman

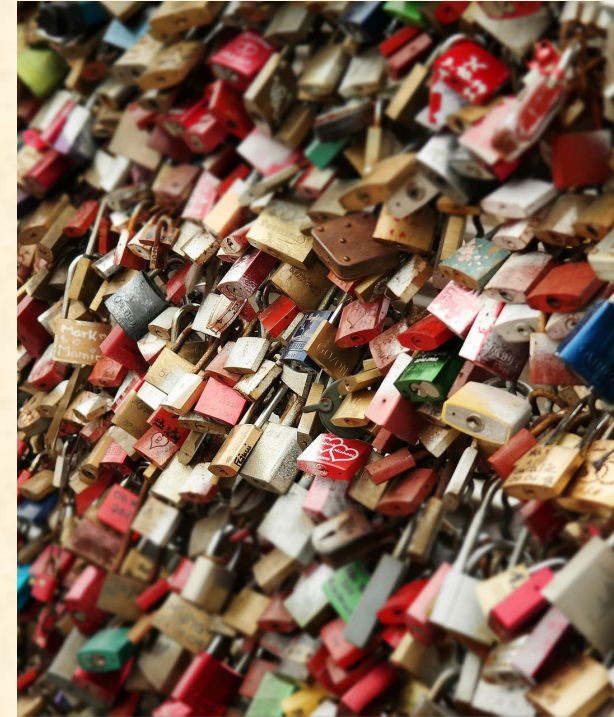
Security Mitigation & Validation

“The trouble with programmers is that you can never tell what a programmer is doing until it’s too late.”

– *Seymour Cray*

■ Anti-Patterns for security mitigation & validation

- Poorly considered password policy
- Poorly considered privilege management
- Assuming firewall or air gap is perfect security
- No implementing secure update + secure boot
- Just relying on penetration testing



■ Mitigation best practices

- Keep up to date with good security practices
- Secure update + secure boot
- Penetration testing is only a starting point

Principle: Password Strength

■ Typical failure scenarios

- Same password used by everyone
- Weak passwords (“1234”)
- Strong password policy → post-it note work-around

■ Possible solutions

- Different password per person with reasonable strength
- Two-factor authentication (e.g., RFID transponder)

■ Balance between usability & security

- Can you memorize: 7R#Ve9j3e@ahi7gjHr(*\pW4!X?
- 2017 NIST guidelines (<https://pages.nist.gov/800-63-3/>)
 - Good ideas: long size, hash/salt/stretch for storage
 - Avoid: words in dictionary, requiring weird characters, password hints, timed expiry
 - Avoid SMS for 2fa (!) due to phone number hijacking (at least in some countries)



B055man69

Storing Passwords

- Don't store them as plain text!
 - Don't just encrypt them either
- Hash:
 - Store a digest of password
 - But, dictionary attacks are a problem
 - Rainbow table: precomputed hashes
- Salting & pepper:
 - Salt: random extra text
 - Pepper: systematic extra text
 - Can be secret or public (tradeoffs)
- Generically, key stretching:
 - E.g., PBKDF2 stretching
 - Use up to date techniques!

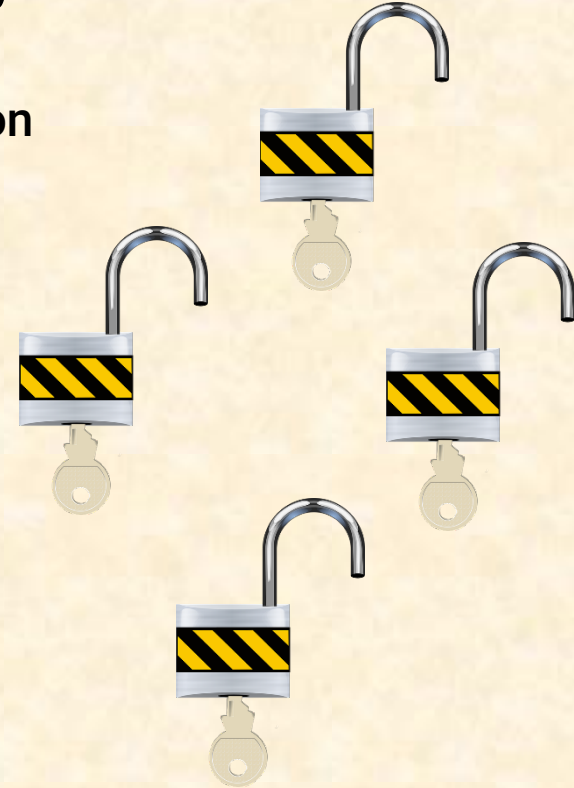
HACKERS RECENTLY LEAKED **153 MILLION** ADOBE USER EMAILS, ENCRYPTED PASSWORDS, AND PASSWORD HINTS. ADOBE ENCRYPTED THE PASSWORDS IMPROPERLY, MISUSING BLOCK-MODE 3DES. THE RESULT IS SOMETHING WONDERFUL:

USER	PASSWORD	HINT	
4e18acc1ab27a2d6		WEATHER VANE SWORD	<input type="text"/>
4e18acc1ab27a2d6			<input type="text"/>
4e18acc1ab27a2d6	a0a2876eb1ea1fca	NAME 1	<input type="text"/>
8babb6299e06eb6d		DUH	
8babb6299e06eb6d	a0a2876eb1ea1fca		<input type="text"/>
8babb6299e06eb6d	85e9da81a8a78adc	57	
4e18acc1ab27a2d6		FAVORITE OF 12 APOSTLES	
1ab29ae86dab6e5ca	7a246a0a2876eb1e	WITH YOUR OWN HAND YOU HAVE DONE ALL THIS	
a1f9b2b6299e7a2b	endcode1e6ab797397	SEXY EARLOBES	<input type="text"/>
a1f9b2b6299e7a2b	617ab0217727ad85	BEST TOS EPISODE	<input type="text"/>
3973867adb06ba7	617ab0217727ad85	SUGARLAND	
1ab29ae86dab6e5ca		NAME + JERSEY #	
877ab7889d3862b1		ALPHA	<input type="text"/>
877ab7889d3862b1			<input type="text"/>
877ab7889d3862b1			<input type="text"/>
877ab7889d3862b1		OBVIOUS	<input type="text"/>
877ab7889d3862b1		MICHAEL JACKSON	<input type="text"/>
38a7c9279codeb44	9dca1d79d4dec6d5		
38a7c9279codeb44	9dca1d79d4dec6d5	HE DID THE MASH, HE DID THE PURLOINED	<input type="text"/>
38a7c9279codeb44			<input type="text"/>
a8e574562b7af7a	9dca1d79d4dec6d5	FAV. WATER-3. POKEMON	

THE GREATEST CROSSWORD PUZZLE
IN THE HISTORY OF THE WORLD 2013

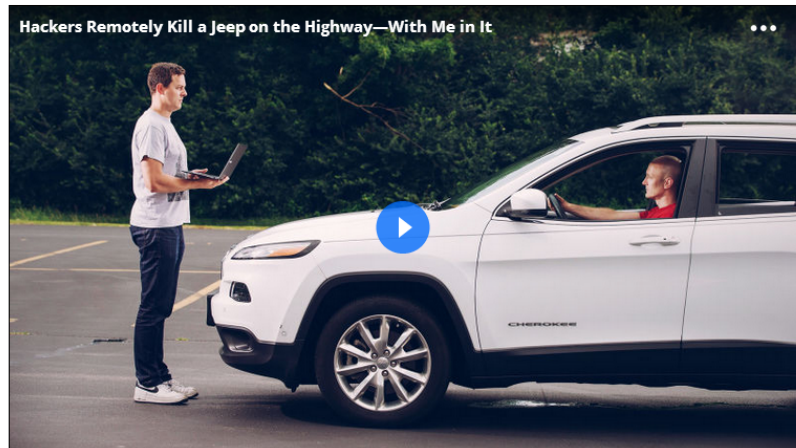
Principle: Least Privilege

- Each user & task should only have as much capability as it needs
 - Commonly, “user,” “administrator,” “factory”
 - Better: per-user fine-grain bit map of function permission
 - Related: helpful to log who did what (forensics)
- Common mistakes
 - Make a common task high privilege
 - Everyone used to log in as admin for Windows
 - Give everyone the same password
 - Once someone has admin, can't roll them back
 - Make risky operations too easy (no confirmation)
- In general, think through permissions
 - Customers may push back, but this is important



What Happens With Unsigned Updates

HACKERS REMOTELY KILL A JEEP ON THE HIGHWAY—WITH ME IN IT



SHARE

SHARE 202660

TWEET 23228

FIN 186

COMMENT 717

EMAIL

I WAS DRIVING 70 mph on the edge of downtown St. Louis when the exploit began to take hold.

Though I hadn't touched the dashboard, the vents in the Jeep Cherokee started blasting cold air at the maximum setting, chilling the sweat on my back through the in-seat climate control system. Next the radio switched to the local hip hop station and began blaring Skee-lo at full volume. I spun the control knob left and hit the power button, to no avail. Then the windshield wipers turned on, and wiper fluid blurred the glass.

As I tried to cope with all this, a picture of the two hackers performing these stunts appeared on the car's digital display: Charlie Miller and Chris Valasek, wearing their trademark track suits. A nice touch, I thought.

LATEST NEWS

CELEBRATIONS
Wiseley's Cool Space Book Comes Personalized to Your...
45 MINS

ANIMAL SCIENCE
Aburd Creatures: Bats Are Totally Legit. Trust Me, I'm a Vampire
2 HOURS

BRANDED CONTENT
How the Adventure Capitalist Gets More Out of Every Trip
THE MARRIOTT REWARDS PREMIER CREDIT CARD

■ Infotainment-to-CAN Firewall CPU non-secured update

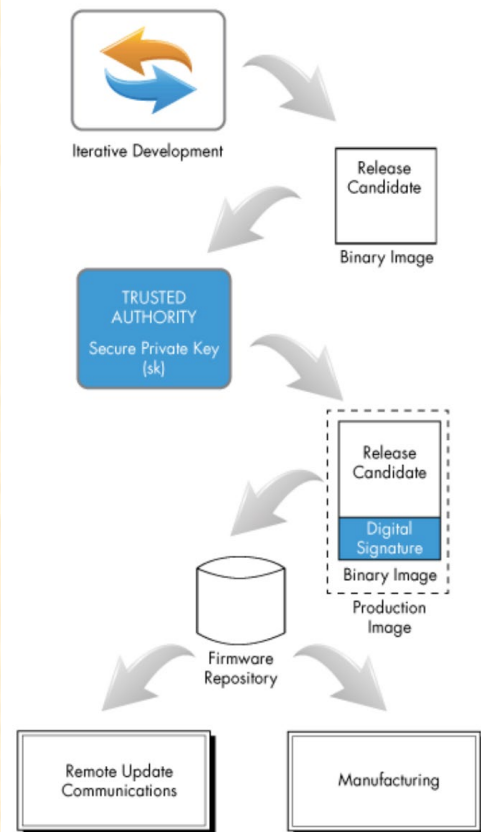
- Attackers reflashed firewall to access CAN



<http://illmatics.com/Remote%20Car%20Hacking.pdf>

Secure Update

- You'll need to deploy security patches
 - Your code might have a vulnerability
 - 3rd party code (library, OS, communications) might be vulnerable
- Secure update good practices:
 - Bootloader that does updates
 - First stage: integrity check for 2nd stage; *can't* be changed(!)
 - Second stage: knows how to load application image
 - Bootloader checks image public key signature
 - Public key hard-coded into bootloader
 - Only properly signed images are loaded
 - Consider limited date ranges (key revocation is hard)
 - » E.g., pre-deploy public key every 3 months for 20 years
 - Consider hard-coding repository IP addresses



Example Mitigation: Secure Boot

- If your firmware is compromised, you are insecure
 - Need a way to make sure you only run factory-authorized code
 - Use public key signature to check firmware image integrity
 - Note: symmetric hash exposes signing key to attack

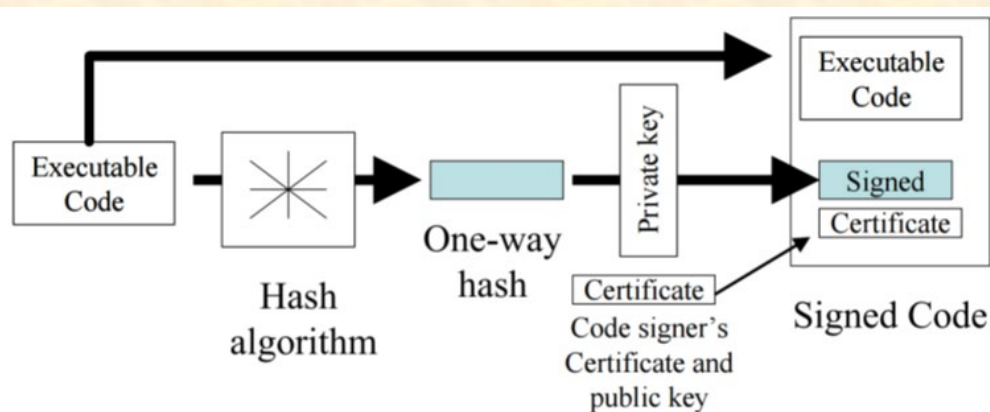


Figure 24. Code- and Document-Signing Process

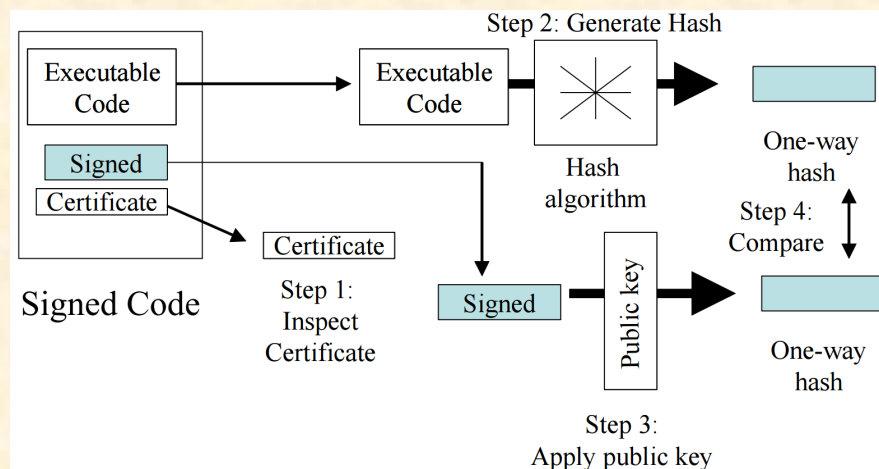


Figure 25. Code- and Document-Signing Verification Process

Encrypting vs. Signing



■ Misconception: “Encryption Equals Security”

- Encryption provides secrecy – but you might need integrity!
- Encryption invokes export controls
- What are the actual security requirements?

■ Example for firmware distribution

- Symmetric key encryption of firmware is a bad idea
 - Key recovery permits adversary to sign malicious images
- Public key encryption of firmware addresses secrecy
 - Reverse engineering will recover firmware image and/or decrypt key
 - But strong crypto secrecy tends to invoke export controls!
- **Secure signature (Public Key Digest) works well**
 - A digest is a small hash of the entire message (like a checksum, but crypto-secure)
 - Sign image off-line one time; all devices can use public key to validate
 - Use per-download encryption as defense in depth

“Pen test” – attempt to attack system to look for problems

■ Automated vulnerability testing

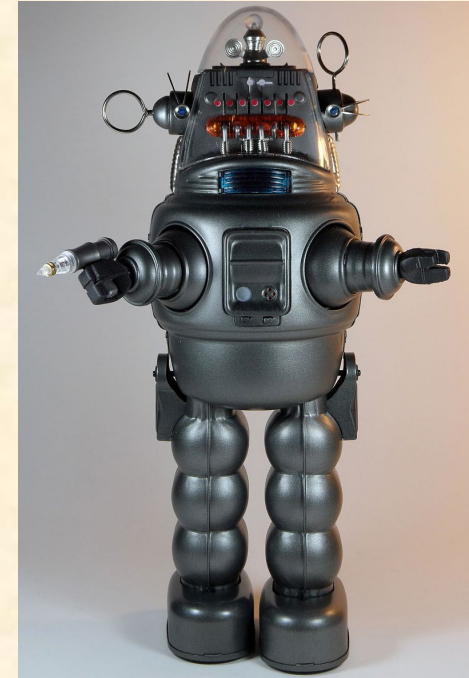
- Test known security exploits to see if they succeed
- Test for bug fixes for known non-exploited bugs
- Port scanning for dangerous open (unnecessary) Ethernet ports

■ Penetration analysis

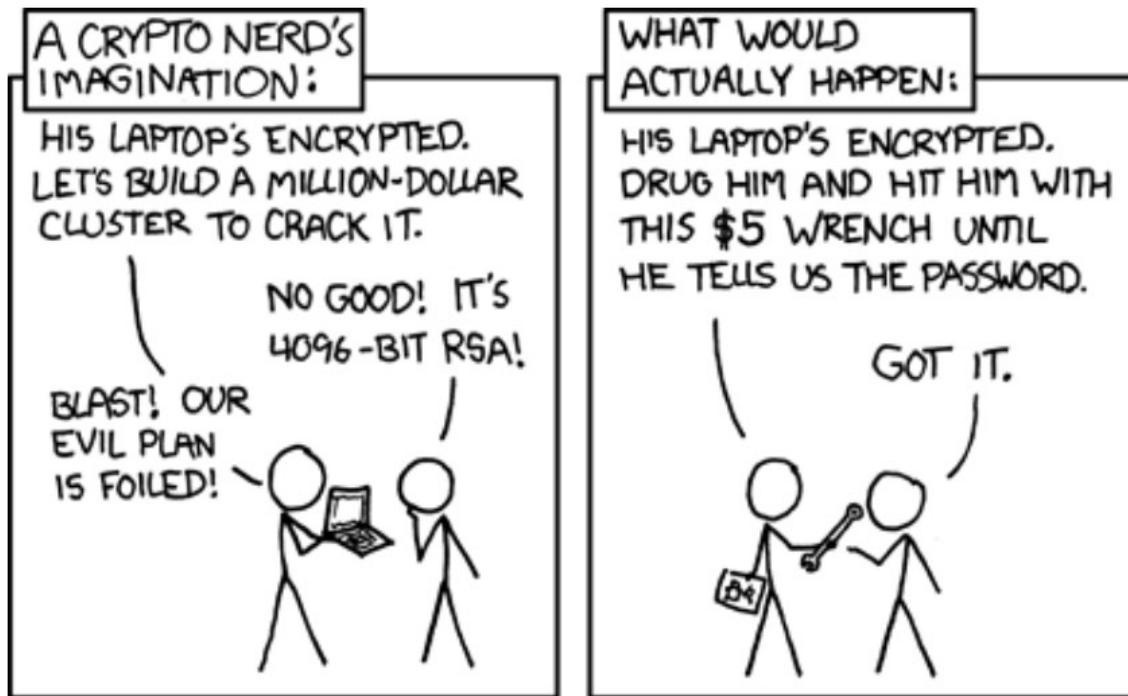
- Hire a “red team” to attempt to penetrate system
- Fuzz testing – send random inputs; see what breaks

■ Looks for likely-to-be-exploited vulnerabilities

- Does not guarantee perfect security



See Also: "Rubber Hose Attack"



■ Static & dynamic code analysis

- General code quality tools: Coverity, PC-Lint
- Security-specific security tools
 - Look for violations of checkable secure coding rules
 - Various tools for thread safety, bounds checking, ...
- Potential problem:
 - False positives (many warnings are not actual vulnerabilities)



■ Peer review

- Security-oriented review of source code
- E.g., Cert C 98 Coding Standard
 - <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1255.pdf>
 - E.g., use `strcpy_s()` instead of `strcpy()`

Many Other Approaches

■ Intrusion detection

- Detect abnormal patterns of system operation
- False positives are expensive; no such system is perfect

■ Monitor Black Hat sites

- Look for published exploits against your product

■ Honey pot systems

- Deploy a monitored decoy system and look for successful attacks

■ Bug bounties

- Pay anyone who finds an exploit so you can fix it



■ Good practices:

- Encourage strong but usable passwords
- Use fine-grain permissions
- Be careful storing password information
- Respect limitations of firewall approaches
- Use secure update and secure boot
- Use more than just penetration testing



■ Pitfalls:

- Thinking security is easy
- Using intuition instead of doing your homework

HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.



OH, DEAR - DID HE BREAK SOMETHING?
IN A WAY -



DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?



OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.

WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.



AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

<https://xkcd.com/327/>